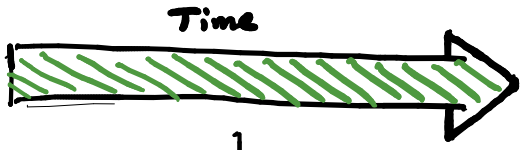
WHEN?



Ancient Romans

Caesar used codes with his generals

Nazi Germany

Alan Turing cracked U-Boat codes.

Modern Security

Governments and corporations want to protect their interests

WHY?

In short: you want to protect your data!

CRYPTOGRAPHY

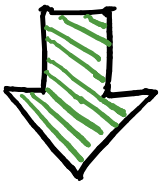
What are...

SECRET KEYS?

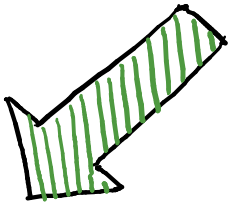
- A secret key is a piece of code that is used to encrypt and/or decrypt messages.

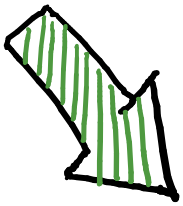
  → think of it like a traditional door and key: its only useful when you have the only copy!

However, you can't just choose any arbitrary key to encrypt data.



Too small and it can be brute-forced by a list of possibilities. Too large and the key will become too cumbersome.

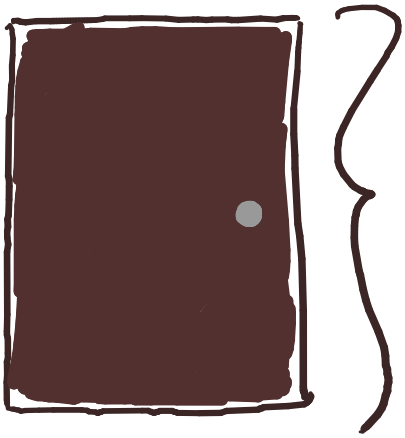




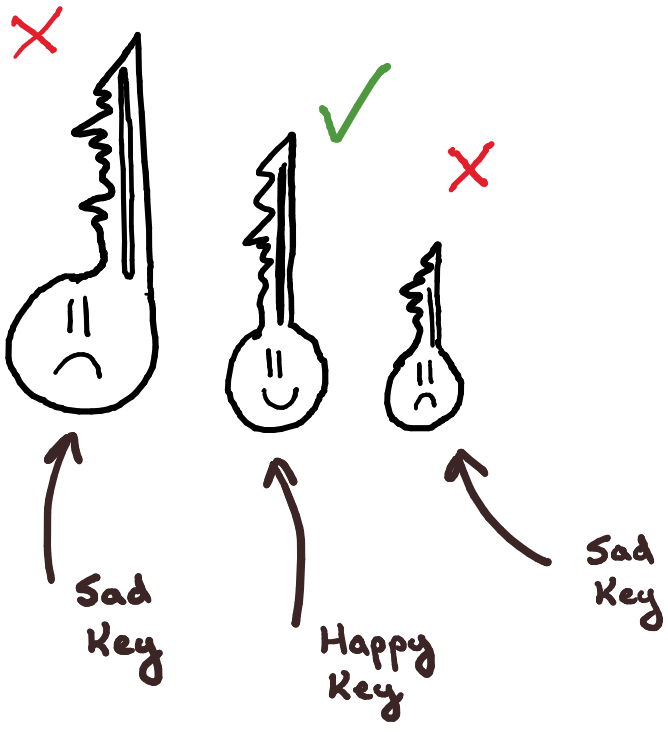
SIZE vs. EFFECTIVENESS

The "key" here is to select a key of moderate size.

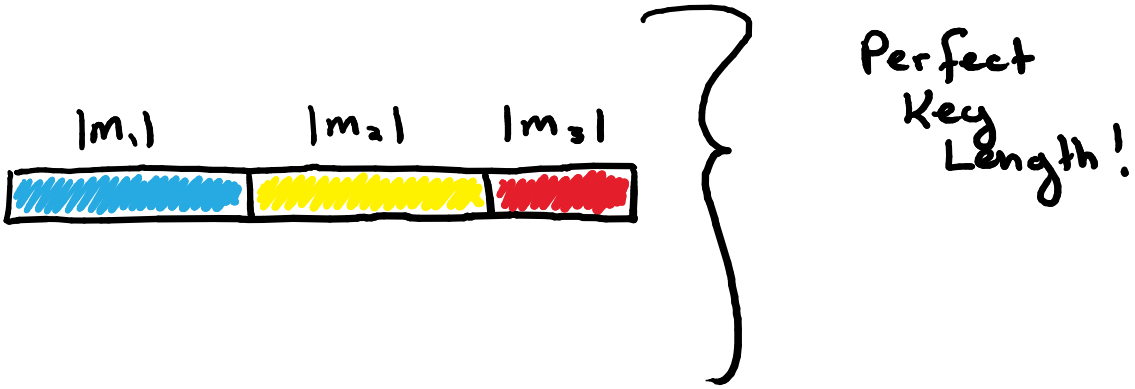
Recall the traditional door and key:



Would you carry a key with just one notch? What about a key three feet long? Probably not; neither extremes are feasible!



The perfect key length is the size of all the messages combined.



A key of this size is called a one-time-pad.

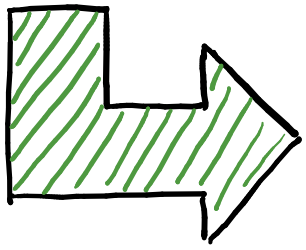
BUT...

... what if we wanted to send a lot of messages? Then, a one-time-pad would be huge, and therefore, ineffective!

↳ Theoretically we want a key that's moderately sized and allows for unlimited communication. However, life isn't fair: a key like this cannot exist!

NOW WHAT?

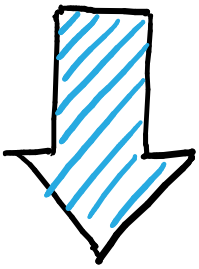
You may have noticed that we can't really create a good key just by choosing a length. However, the advancement of complexity theory allows us to create a code based on a problem that we think is hard to solve.



Hooray! Our data is secure! But what we want to communicate with different people?

PUBLIC KEY vs. PRIVATE KEY

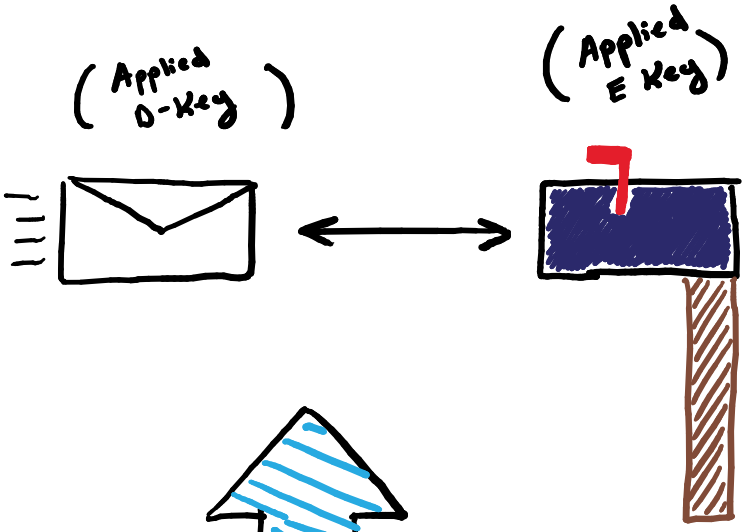
- Encryption key is different from the decryption key.
- The encryption key is public and available.
- The decryption key is secret and private

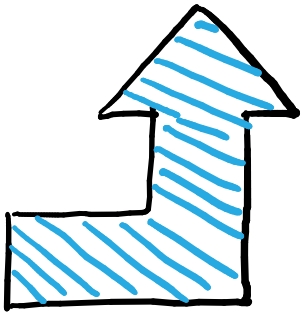


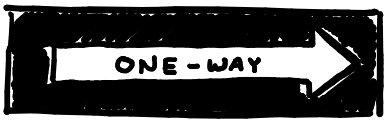
We can also use public key cryptosystems for digital signatures

By applying the decryption algorithm before sending a message, the recepient can verify the sender by applying the encryption algorithm.

- Encryption and decryption keys are identical.
- The key must be secret!





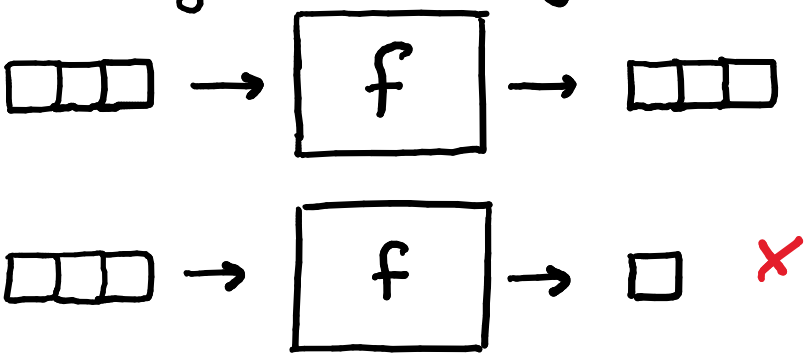


FUNCTION

This is the theory behind modern private-key cryptosystems!

First, we need some background:

① A length-preserving function



Formally: $f: \Sigma^* \to \Sigma^*$

Basically, a function $f(w)$ outputs the same length as input w.

② Probabilistic function

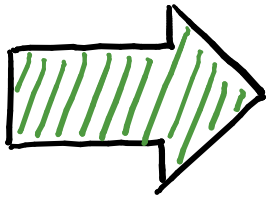
Formally:

$M: \Sigma^* \to \Sigma^*$

$Pr(M(w) = x)$

This outputs the probability that a machine M accepts (halts) on x, starting with input w.

THE DEFINITION:



Continued!

One-way Permutation

① Computable in polynomial time.

② For every probabilistic TM M, $\forall k$ and large n, if we pick a random w with length n:

$$Pr_{m,w}\left[M(f(w)) = w\right] \leq n^{-k}$$

One-way Function

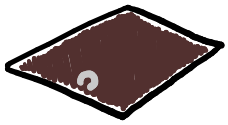
1. Computable in polynomial time.

2. For every probabilistic TM M, $\forall k$ and large n, if we pick a random w with length n:

$$Pr\left[M(f(w)) = y, \text{ where } f(y) = f(w)\right] \leq n^{-k}$$

☆ Both of these functions are length-preserving.

TRAPDOOR FUNCTION



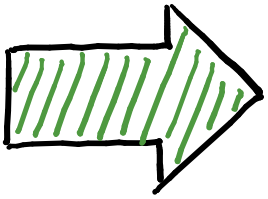
This is the theory behind modern public-key cryptosystems!

First, some background:

1. Indexing Function

If we have a group of functions $\{f_i\}$ for each $i \in \Sigma^*$, we can represent them with:

$$f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$



THE DEFINITION:

Continued!

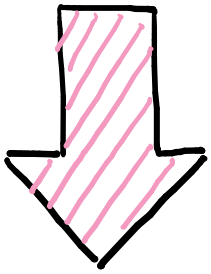
(1) Functions f and h are computable in polynomial time.

(2) For every probabilistic time TM, E, and $\forall k$ and large n, if we take a random output $\langle i, t \rangle$ of G on 1^n and a random $w \in \Sigma^n$, then:

$$Pr\left[E(i, f(w)) = y, \text{ where } f_i(y) = f_i(w)\right] \leq n^{-k}$$

(3) For every n, every w of length n, and every output $\langle i, t \rangle$ of G that occurs with non-zero probability for some input to G:

$$h(t, f_i(w)) = y, \text{ where } f_i(y) = f_i(w)$$



We can use this to create a public-key cryptosystem:

We generate a public key i, by the machine G. The value of t is the secret key. The algorithm takes a message m, breaks it into blocks no larger than $\log(N)$. For each of these blocks, we compute f_i.

The resulting sequence of strings is the encrypted message. The receiver uses the function h to obtain the original message from its encryption.